Cache Me If You Must: Adaptive Key-Value Quantization for Large Language Models

Alina Shutova* [1] Vladimir Malinovskii* [1,2] Vage Egiazarian* [3]
Denis Kuznedelev [2] Denis Mazur [4, 5] Nikita Surkov [6]
Ivan Ermakov [5] Dan Alistarh [3]

[1]HSE University [2]Yandex Research [3]IST Austria [4]SberDevices [5]MIPT
[6]T-Bank

The KV Cache Bottleneck

Key-Value (KV) caches enable efficient LLM decoding but consume tens of GBs for long contexts. Existing compression methods, such as pruning or quantization sacrifice quality at high compression rates.

Our method—AQUA-KV:

- combines quantization with using **inter- and intra-layer dependencies** between keys and values;
- shows near-lossless quality at **2-2.5 bits** per value with **under 1% relative error** in perplexity and LongBench score;
- can be **calibrated within 1-6 hours** for 70B models.

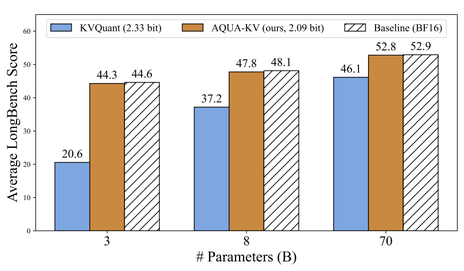

Figure 1. Comparison of AQUA-KV to an alternative Key-Value Cache compression method KVQuant in near-2bit and the full-precision model on Llama 3 model family.

Key insights

1. **Inter-layer dependencies:** keys and values in adjacent layers exhibit strong correlations.

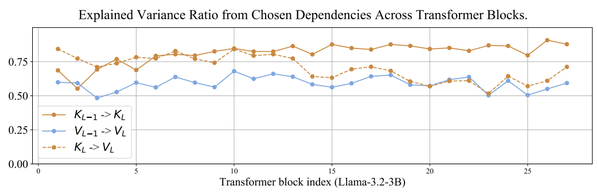

Figure 2. Explained Mean Variance Ratios per Transformer Block for chosen sets of linear probes on Llama-3.2-3B.

2. **Intra-layer dependencies:** within each layer, keys and values are highly interlinked.

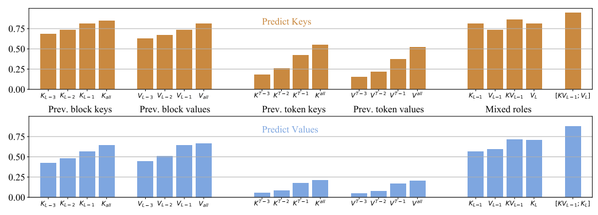

Figure 3. Mean Explained Variance Ratios by linear probes from previous blocks (L), tokens (T) and role on Llama-3.2-3B.

Method

Dependency-aware compression:

- Compact **linear** adapters predict reusable KV vectors across layers
- Quantizes **only residuals** for minimal information loss
- Best results with data-free vector quantization for residuals

Hardware-aware design:

- **One-shot** calibration
- Compatible with any quantization method
- Complementary to pruning
- Can be used with any Transformer architecture

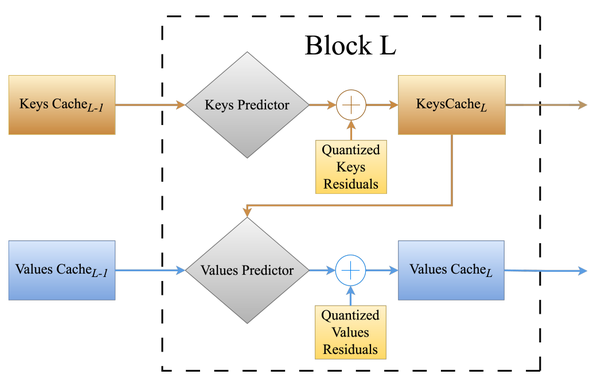

Figure 4. AQUA-KV inference scheme.

Experiments

We evaluate our method on models from the Llama 3 [3] and Qwen 2.5 LLM [8] families and report PPL on WikiText-2 and mean LongBench v1 [2] score.

AQUA-KV improves the quality of all of the chosen backbone quantization algorithms and outperforms known KV-cache compression methods in 2-bits:

Table 1. Evaluation of Llama 3.2 3B with various Key-Value cache compression strategies.

Method	Quant. Bits	Wiki2 PPL↓ (base model)	LongBench Avg.↑ (instruct model)
Uncompressed	16	6.98	44.47
Quanto [5]	2.50	21.56	33.59
KIVI [6]	2.25	9.33	39.63
KVQuant [4]	2.33	9.43	20.56
QuaRot [1]	2.03	44.68	32.80
HIGGS [7]	2.02	7.47	43.25
AQUA-KV (Quanto)	2.64	10.33	43.64
AQUA-KV (QuaRot)	2.17	8.44	**44.54**
AQUA-KV (HIGGS)	2.16	**7.03**	44.26

The usage of pruning together with AQUA-KV can grant up to 40× compression with the quality compatible to the pruning itself.

Table 2. Evaluation of AQUA-KV quantization in combination with H_2O token pruning on Llama 3. Every entry uses H_2O procedure to keep only 20% tokens, using hyperparameters from [9].

Method	Quant. Bits	Memory Saved	LongBench Avg.↑ 3B	8B
H_2O only	16	5.0×	38.82	41.42
H_2O + HIGGS	2.02	39.6×	37.02	40.72
H_2O + AQUA-KV	2.09	38.3×	38.43	41.11

Experiments

AQUA-KV's 2-bit compression outperforms HIGGS and prior work, often matching 3-bit baselines despite using **32% fewer bits**.

Table 3. Evaluation of AQUA-KV (with HIGGS backbone) and baselines across five LLM for 2 & 3 bit compression. The cache size shows total memory footprint for Llama 3.1 70B model with sequence length 131k tokens and batch size 1, *including predictors*.

Method	Quant. bits	WikiText-2 PPL↓ Llama 3.x 3B	8B	70B	LongBench AVG↑ (instruct) Llama 3.x 3B	8B	70B
–	16	6.98	5.61	2.54	44.61	48.13	52.92
AQUA-KV	2.09	**7.03**	**5.72**	**2.62**	**44.30**	**47.77**	**52.79**
HIGGS	2.02	7.47	5.89	2.77	42.80	47.37	52.18
KIVI	2.25	9.34	7.37	3.06	39.64	46.28	52.45
KVQuant	2.33	9.43	6.64	3.28	20.56	37.17	46.14
AQUA-KV	3.06	**6.98**	**5.64**	**2.55**	44.37	**48.10**	52.81
HIGGS	3.02	7.05	5.66	2.57	**44.41**	47.86	52.56
KIVI	3.05	7.87	6.04	2.87	41.40	46.98	**52.87**
KVQuant	3.33	7.26	5.84	2.75	41.40	46.42	50.74

Inference Efficiency

AQUA-KV's custom CUDA kernels achieve near-native latency (18% overhead at 32K context) while boosting throughput by 35% via 60% KV cache reduction, outperforming naive PyTorch by 3.2×.

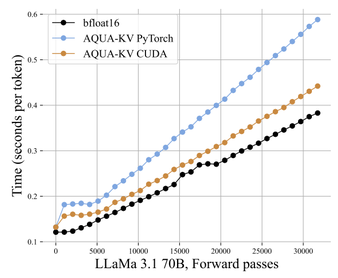

Figure 5. Inference latency with batch size 1 for Llama 3.1 70B on A100 GPU

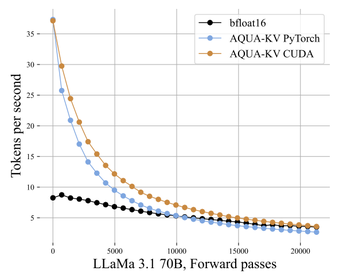

Figure 6. Throughput of batched inference for Llama 3.1 70B on A100 GPUs

References

[1] Saleh Ashkboos et al. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024.

[2] Yushi Bai et al. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

[3] Abhimanyu Dubeyet al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[4] Connor Hooper et al. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv:2401.18079*, 2024.

[5] HuggingFace. Optimum-quanto: A pytorch quantization backend for optimum. `https://github.com/huggingface/optimum-quanto`, 2024. Accessed: 2025-01-28.

[6] Zechun Liu et al. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024.

[7] Vladimir Malinovskii et al. Pushing the limits of large language model quantization via the linearity theorem. *arXiv preprint arXiv:2411.17525*, 2024.

[8] Qwen Team. Qwen2.5: A party of foundation models, September 2024.

[9] Zhenyu Zhang et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

ICML 2025, 13-19 July 2025, Vancouver, Canada